978-1-4673-1228-8/12/$31.00 © 2012 IEEE

Autonomous Cleaning of Corrupted Scanned Documents – A Generative Modeling Approach

Zhenwen Dai
Frankfurt Institute for Advanced Studies,
Dept. of Physics, Goethe-University Frankfurt
dai@fias.uni-frankfurt.de

Jörg Lücke
Frankfurt Institute for Advanced Studies,
Dept. of Physics, Goethe-University Frankfurt
luecke@fias.uni-frankfurt.de

Abstract

We study the task of cleaning scanned text documents that are strongly corrupted by dirt such as manual line strokes, spilled ink etc. We aim at autonomously removing dirt from a single letter-size page based only on the information the page contains. Our approach, therefore, has to learn character representations without supervision and requires a mechanism to distinguish learned representations from irregular patterns. To learn character representations, we use a probabilistic generative model parameterizing pattern features, feature variances, the features' planar arrangements, and pattern frequencies. The latent variables of the model describe pattern class, pattern position, and the presence or absence of individual pattern features. The model parameters are optimized using a novel variational EM approximation. After learning, the parameters represent, independently of their absolute position, planar feature arrangements and their variances. A quality measure defined based on the learned representation then allows for an autonomous discrimination between regular character patterns and the irregular patterns making up the dirt. The irregular patterns can thus be removed to clean the document. For a full Latin alphabet we found that a single page does not contain sufficiently many character examples. However, even if heavily corrupted by dirt, we show that a page containing a lower number of character types can efficiently and autonomously be cleaned solely based on the structural regularity of the characters it contains. In different examples using characters from different alphabets, we demonstrate generality of the approach and discuss its implications for future developments.

1. Introduction

A basic form of human communication, written text, consists of planar arrangements of reoccurring and regular patterns. While in modern forms of text these patterns are characters or symbols for words (e.g., Chinese texts), early forms consisted of symbols resembling objects. Written text became a successful form of communication because it exploits the readily available capability of the human visual system to learn and recognize regular patterns in visual data. In recent years, computer vision and machine learning became increasingly successful in analyzing visual data. Much progress has been made, for instance, by probabilistic modeling approaches that aim at capturing the statistical regularities of a given data set. Examples are image denoising by Markov Random Fields [1] or sparse coding models [2, 3]. For many types of data, modeling approaches hereby have to address the problem that regular visual structures often appear at arbitrary positions. Sparse coding approaches indirectly address this problem by replicating a learned structure (e.g., a Gabor wavelet) at different positions of image patches. Other approaches go one step further and explicitly model pattern positions using additional hidden variables [4, 5, 6, 7, 8, 9]. However, the combinatorics of object identity and position introduces major challenges as for each pattern class all positions ideally have to be considered.

In this paper we apply a probabilistic generative approach with explicit position encoding to remove dirt from text documents. The principle idea is very straight-forward: If characters are the salient regular patterns of text, an appropriately structured probabilistic model should be able to learn character representations as regular arrangements of features. In contrast, dirt is much more irregular. Coffee spots, spilled ink, or line-strokes scratching-out text share similar features with printed characters but such corruptions are, on average, much more random combinations of feature patterns. Based on this observation, the autonomous identification and recovery of characters from a corrupted text document should thus be possible. But how difficult is such a task? Or how robust can a solution of such a task be if the data is heavily corrupted by dirt? Would the information contained on a single page of a dirty document, for instance, be sufficient to identify the characters con-

taining it? And if yes, can this be used to 'self-clean' the document? Such questions can, of course, not be answered by a clear 'yes' or 'no' because they will, e.g., depend on the type and degree of dirt or on the amount of available character information on a page. However, we will show that a self-cleaning of heavily corrupted documents is, indeed, possible, e.g., for relatively low numbers of different character types. The only prerequisite will hereby be the characters' regular feature arrangements. No information about the characters has to be available, which makes the approach applicable to entirely unknown character types. The problem addressed here is thus very different from the one aimed at by optical character recognition (OCR) methods that use supervised pretraining on known characters. In contrast, we require unsupervised methods to learn character representations. The generative model we apply is similar to models suggested by Williams & Titsias [8] and Jojic & Frey [6, 9, 10], which provide explicit representations of the data's regular patterns. As the data points we will have to process are image patches of corrupted text documents, these previous models are not applicable because they require a static background, do not provide a mechanism to discriminate characters from irregular patterns, and are based on pixel image representations which can make learning less robust. In contrast, we (1) will have to allow for varying fore- and background patterns (to take dirt into account), (2) will introduce a mechanism for character vs. dirt discrimination, and (3) will consider general feature vector representations of the data. Together with a novel non-greedy training scheme in the form of truncated variational EM [11], the derived method will provide the required robustness and efficiency for the task.

2. A Generative Model for Characters

The probabilistic model we consider generates small image patches of size $\vec{D} = (D_1, D_2)$. A pixel at position $\vec{d}$ of the patch is represented by a feature vector $\vec{y}_{\vec{d}}$ with F entries. For now $\vec{y}_{\vec{d}}$ can be thought of as a color vector at pixel position $\vec{d}$ in RGB space ($F = 3$). For the application to text documents we will later use more sophisticated features, however.

A patch $Y = (\vec{y}_{(1,1)}, \ldots, \vec{y}_{(D_1,D_2)})$ is modeled to contain one pattern at an arbitrary position of the patch (see Fig. 1a). For the class variable c we use a standard mixture model prior with $\vec{\pi} = (\pi_1, \ldots, \pi_C)$ denoting the mixing proportions and C denoting the total number of classes:

$$p(c|\vec{\pi}) = \pi_c \qquad \text{with} \quad \textstyle\sum_{c=1}^{C} \pi_c = 1\,. \tag{1}$$

The position of the pattern in the patch, $\vec{x} \in \mathcal{D}, \mathcal{D} = \{1, \ldots, D_1\} \times \{1, \ldots, D_2\}$, is a 2D vector chosen from a uniform distribution over the entire patch:

$$p(\vec{x}) = p(x_1)p(x_2) = \text{Uniform}(1, D_1) \times \text{Uniform}(1, D_2) = \frac{1}{D_1 D_2}. \tag{2}$$

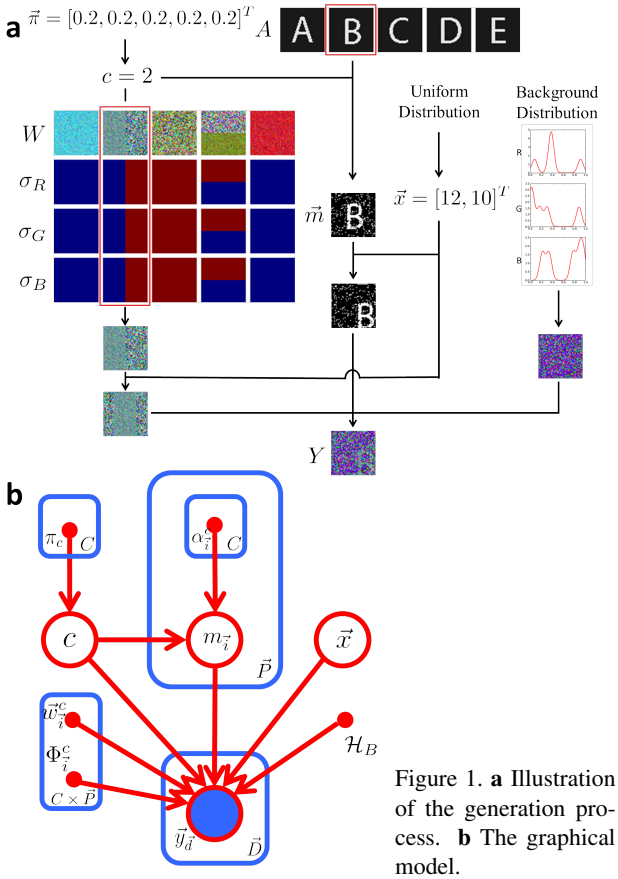

Figure 1. **a** Illustration of the generation process. **b** The graphical model.

The shapes of different patterns are modeled by a set of binary latent variables, namely the pattern *mask*: $\vec{m} = (m_{(1,1)}, \ldots, m_{(P_1,P_2)})$, where $m_{\vec{i}} \in \{0, 1\}$. With $m_{\vec{i}} = 1$ the corresponding feature is part of the pattern, while with $m_{\vec{i}} = 0$ it is part of the background. The pattern size $\vec{P} = (P_1, P_2)$ can be different from the image patch size $P_1 \leq D_1, P_2 \leq D_2$. Given the pattern class c, the mask variables are drawn from Bernoulli distributions:

$$p(\vec{m}|c, A) = \prod_{\vec{i}=(1,1)}^{(P_1,P_2)} p(m_{\vec{i}}|c, A) = \prod_{\vec{i}=(1,1)}^{(P_1,P_2)} (\alpha^c_{\vec{i}})^{m_{\vec{i}}} (1 - \alpha^c_{\vec{i}})^{1-m_{\vec{i}}}, \tag{3}$$

where $A = (A^1, \ldots, A^C)$ with $A^c = (\alpha^c_{(1,1)}, \ldots, \alpha^c_{(P_1,P_2)})$ are the parameters of the mask distribution. For the area where the image patch is outside the pattern, the mask variables are always assigned 0: $p(m_{\vec{d}} = 1 \,|\, c, A) = p(m_{\vec{d}} = 1) = 0, \forall \vec{d} \in \mathcal{D} - \mathcal{P}, \mathcal{P} = \{1, \ldots, P_1\} \times \{1, \ldots, P_2\}$. From the definition of masks, a background distribution is required for all those features not belonging to a pattern ($m_{\vec{i}} = 0$). A possible choice is a flat Gaussian distribution (compare [8]). However, for data such as patches from corrupted text documents, the distribution values are often very different for the different feature vector entries, and for the dirty background are often observed to be non-Gaussian. To appropriately model

the background features, we therefore construct a probability density function $\mathcal{H}_B$ by computing the histogram of different feature values across the image patches. The probability densities for the individual feature vector entries will be modeled individually (see Fig. 2a for histograms of R, G, and B channel). The histograms are computed across all the image patches including the features that are potentially later identified as being part of the learned patterns. Nevertheless, the computed histograms are usually very similar to the true background distributions (compare Fig. 2a). Once computed we therefore leave the histograms fixed throughout learning. Having defined the background distribution $\mathcal{H}_B$ and given pattern class c, mask $\vec{m}$, and pattern position $\vec{x}$, the distribution of patch features is given by:

$$p(Y|c,\vec{m},\vec{x},\Theta) = \prod_{\vec{d}=(1,1)}^{(D_1,D_2)} \Big[m_{(\vec{d}-\vec{x})} \mathcal{N}(\vec{y}_{\vec{d}}\,; \vec{w}^c_{(\vec{d}-\vec{x})}, \Phi^c_{(\vec{d}-\vec{x})}) + (1 - m_{(\vec{d}-\vec{x})}) \mathcal{H}_B(\vec{y}_{\vec{d}}) \Big], \tag{4}$$

where $\vec{w}^c_{\vec{i}}$ is the mean of a Gaussian distribution and $\Phi^c_{\vec{i}}$ is the diagonal convariance matrix: $\Phi^c_{\vec{i}} = \mathrm{diag}\big((\sigma^c_{\vec{i},f=1})^2, \ldots, (\sigma^c_{\vec{i},f=F})^2\big)$. The mean $\vec{w}^c_{\vec{i}}$ parameterizes the mean feature vector of pattern c at position $\vec{i}$ relative to the pattern position $\vec{x}$. The variance vector $\Phi^c_{\vec{i}}$ parameterizes the feature vector variances (different variance per vector entry). The shift of a pattern c is implemented by a change of the position indices $\vec{i}$ by $\vec{x}$ using cyclic boundary positions:

$$\vec{d} = (\vec{i}+\vec{x}) := \Big((i_1 + x_1) \bmod D_1, (i_2 + x_2) \bmod D_2 \Big)^T. \tag{5}$$

Equations 1 to 5 define the generative model for image patches. The parameters of the model are given by $\Theta = (W, \Phi, A, \vec{\pi})$ with $W = (W^0, \ldots, W^C)$ and $W^c = (\vec{w}^c_{(1,1)}, \ldots, \vec{w}^c_{(P_1,P_2)})$, and together with the histograms for the background distribution. Fig. 1a shows schematically how a patch is generated. First, a pattern class is chosen (e.g., the class with pattern "B") and then the mask variables $\vec{m}$ for the class (Eqn. 3). Pattern parameters and mask are then translated by a random position $\vec{x}$ before they are combined through a Gaussian distribution for the model and the learned distribution for the background Eqn. 4.

3. Efficient Likelihood Maximization

One approach of learning the parameters Θ from data $\mathcal{Y} = (Y^{(1)}, \ldots, Y^{(N)})$ is to maximize the data likelihood:

$$\Theta^* = \arg\max_{\Theta} \{\mathcal{L}(\Theta)\}, \qquad \mathcal{L}(\Theta) = \log\big(p(Y^{(1)}, \ldots, Y^{(N)}|\Theta)\big). \tag{6}$$

A frequently used method to find the parameters Θ^* is Expectation Maximization (EM), which iteratively optimizes a lower bound of the likelihood $\mathcal{F}(\Theta, q)$ w.r.t. the parameters Θ and a distribution q. With $\sum_V$ denoting a summation across the joint space of all hidden variables $V = (c, \vec{m}, \vec{x})$ it is given by:

$$\mathcal{F}(\Theta, q) = \sum_{n=1}^{N} \sum_{V} q_n(V, \Theta') \Big[\log(p(Y^{(n)}|V, \Theta)) + \log(p(V|\Theta)) \Big] - \sum_{n=1}^{N} \sum_{V} q_n(V, \Theta') \log(q_n(V, \Theta')). \tag{7}$$

M-Step. Parameter update rules are canonically derived by setting the derivatives of $\mathcal{F}$ w.r.t. the parameters to 0. For the model (1) - (4), we obtain:

$$\begin{aligned}
\pi_c &= \tfrac{1}{N} \sum_n \sum_{\vec{x}} p^{(n)}_\Theta(c, \vec{x}), \\
\alpha^c_{\vec{i}} &= \frac{\sum_n \sum_{\vec{x}} p^{(n)}_\Theta(c,\vec{x}) p^{(n)}_\Theta(m_{\vec{i}}=1|c,\vec{x})}{\sum_n \sum_{\vec{x}} p^{(n)}_\Theta(c,\vec{x})}, \\
\vec{w}^c_{\vec{i}} &= \frac{\sum_n \sum_{\vec{x}} p^{(n)}_\Theta(c,\vec{x}) p^{(n)}_\Theta(m_{\vec{i}}=1|c,\vec{x}) \vec{y}^{(n)}_{(\vec{i}+\vec{x})}}{\sum_n \sum_{\vec{x}} p^{(n)}_\Theta(c,\vec{x}) p^{(n)}_\Theta(m_{\vec{i}}=1|c,\vec{x})}, \\
\Phi^c_{\vec{i}} &= \frac{\sum_n \sum_{\vec{x}} p^{(n)}_\Theta(c,\vec{x}) p^{(n)}_\Theta(m_{\vec{i}}=1|c,\vec{x}) \big((\vec{y}^{(n)}_{(\vec{i}+\vec{x})} - \vec{w}^c_{\vec{i}})(\vec{y}^{(n)}_{(\vec{i}+\vec{x})} - \vec{w}^c_{\vec{i}})^T \odot \mathbb{1} \big)}{\sum_n \sum_{\vec{x}} p^{(n)}_\Theta(c,\vec{x}) p^{(n)}_\Theta(m_{\vec{i}}=1|c,\vec{x})},
\end{aligned} \tag{8}$$

where we abbreviated: $p^{(n)}_\Theta(m_{\vec{i}}|c,\vec{x}) := p(m_{\vec{i}}|Y^{(n)}, c, \vec{x}, \Theta)$, $p^{(n)}_\Theta(c,\vec{x}) := p(c,\vec{x}|Y^{(n)}, \Theta)$, and where $\odot$ denotes pointwise matrix multiplication (in this case with the unit matrix).

E-Step. The crucial and computationally expensive part of EM is the computation of the expectation values w.r.t. the posterior. For each data point, this involves summations of probabilities for all combinations of the hidden variables c, $\vec{m}$ and $\vec{x}$. However, the summation over the latent combinations can be simplified. By exploiting the standard assumption of independent observed variables (compare, e.g., [2, 3]) given the latents (see Eqn. 4), the posterior distribution over $\vec{m}$ can be decomposed into the product of the posteriors over individual binary masks as follows:

$$p(c, \vec{m}, \vec{x}|Y, \Theta) = \Big(\prod_{\vec{i}=(1,1)}^{(P_1,P_2)} p(m_{\vec{i}}|Y, c, \vec{x}, \Theta) \Big) p(c, \vec{x}|Y, \Theta). \tag{9}$$

The posterior distribution over individual binary masks can then be computed as follows:

$$p(m_{\vec{i}}|Y, c, \vec{x}, \Theta) = \frac{p(\vec{y}_{(\vec{i}+\vec{x})}, m_{\vec{i}}|c, \vec{x}, \Theta)}{\sum_{m'_{\vec{i}}} p(\vec{y}_{(\vec{i}+\vec{x})}, m'_{\vec{i}}|c, \vec{x}, \Theta)}. \tag{10}$$

The summation in the denominator can be computed efficiently as it only contains two cases: $m_{\vec{i}} = 0$ and $m_{\vec{i}} = 1$. The posterior distribution over c and $\vec{x}$ can be computed as follows,

$$p(c, \vec{x}|Y, \Theta) \propto \Big[\prod_{\vec{i}=(1,1)}^{(P_1,P_2)} \Big(\sum_{m_{\vec{i}}} p(\vec{y}_{(\vec{i}+\vec{x})}, m_{\vec{i}}|c, \vec{x}, \Theta) \Big) \Big] \cdot p(\vec{x}|\Theta) p(c|\Theta). \tag{11}$$

With such a decomposition (compare [9, 8]), the computational complexity decreases from exponential to polynomial, which makes the computation tractable in principle. However, the computational complexity still grows very fast with the size of patterns and patches, $\mathcal{O}(CD_1D_2P_1P_2)$. For realistic image sizes (e.g., usually hundreds of thousands of pixels), it still exceeds currently available computational resources. To further improve efficiency, we therefore approximate the computation of expectation values using variational EM (e.g., [12]). Source of the large computation is the required evaluation of all possible pattern positions for all classes. To reduce the number of hidden states that have to be evaluated, we apply a recent variational EM approach (Expectation Truncation, [11]) which is well suited for discrete hidden variables. The used approach is not based on the usual factored form of q but on a truncated variational approximation to the posterior. Applied to the posterior (11) it is given by:

$$\begin{aligned} p(c,\vec{x}\,|\,Y^{(n)},\Theta) &\approx q_n(c,\vec{x};\Theta) \\ &= \frac{p(c,\vec{x},Y^{(n)}\,|\,\Theta)}{\sum_{(c,\vec{x})\in\mathcal{K}_n} p(c,\vec{x},Y^{(n)}\,|\,\Theta)}, \forall (c,\vec{x})\in\mathcal{K}_n, \end{aligned} \tag{12}$$

and zero otherwise. The variational distribution q_n approximates the true posterior with high precision if the set $\mathcal{K}_n$ contains those classes and positions that carry most posterior mass for a given data point $Y^{(n)}$. In other words, for a given patch we have to find the most likely pattern classes together with their most likely patch positions in order to obtain a high quality approximation. To achieve this we define a function $\mathcal{S}_\Theta^{(n)}(c,\vec{x})$ that assigns a score to each class and position pair $(c,\vec{x})$:

$$\begin{aligned} \mathcal{S}_\Theta^{(n)}(c,\vec{x}) = &\prod_{\vec{i}'\in\mathcal{P}'_c} \Big[\mathcal{N}(\vec{y}^{(n)}_{(\vec{i}'+\vec{x})}\,;\vec{w}^c_{\vec{i}'},\Phi^c_{\vec{i}'})p(m_{\vec{i}'}=1|\Theta) \\ &+\mathcal{H}_B(\vec{y}^{(n)}_{(\vec{i}'+\vec{x})})p(m_{\vec{i}'}=0|\Theta)\Big] p(\vec{x}|\Theta)p(c|\Theta), \end{aligned} \tag{13}$$

with $\mathcal{P}'_c \subseteq \mathcal{P}$. This scoring (or selection) function (compare [11]) gives high values to all those positions that are consistent with features in the set $\mathcal{P}'_c$. The set $\mathcal{P}'_c$ is in turn defined to contain the λ most reliable features of pattern c. We define these features as those with the highest mask parameters α^c_i. A small number of λ results in a very efficiently computable function $\mathcal{S}_\Theta^{(n)}(c,\vec{x})$. Based on the selection function, we now define the set of most likely class and position pairs to be:

$$\begin{aligned} \mathcal{K}_n = &\{(c,\vec{x})|(c,\vec{x}) \text{ has one of} \\ &\text{the } (K\ C\ D_1\ D_2) \text{ largest values of } \mathcal{S}^{(n)}_{c,\Theta}(\vec{x})\}, \end{aligned} \tag{14}$$

where $K \in [0,1]$ is the fraction of the joint space of all classes and positions (size $C\ D_1\ D_2$).

In principle, the approximation [11] can also be used to constrain the number of states of mask variables. However, the computational gain is negligible as the posterior w.r.t. the mask can be computed efficiently (10). For the approximation, note that λ and K parameterize the accuracy. The higher λ the more reliably is the selection of considered classes and positions, and the higher K the larger is the considered area of the joint class and position space. However, the larger λ and K the higher is the computational cost. For the highest possible value of λ the selection becomes optimal as $\mathcal{S}_\Theta^{(n)}(c,\vec{x})$ becomes proportional to $p(c,\vec{x}\,|\,Y^{(n)},\Theta)$ ($\mathcal{S}_\Theta^{(n)}(c,\vec{x})$ becomes equal to $p(c,\vec{x}\,|\,Y^{(n)},\Theta)p(Y^{(n)}\,|\,\Theta)$ with $p(Y^{(n)}\,|\,\Theta)$ being a constant for the selection). For the highest possible value of K, $K=1$, all positions are considered and the variational distribution (12) becomes equal to the exact posterior. In numerical experiments we found approximations with high accuracy and simultaneously low computational costs by choosing relatively low numbers of λ (e.g., $\lambda = 200$ out of P_1P_2 features) and relatively low fractions of considered joint space (e.g., $K = 0.02$).

4. Learning and Identification of Characters

Equations (8) to (14) define an approximate EM algorithm to learn character representations. These representations will be used to remove dirt from documents as described in this section. Before, we numerically evaluate the learning procedure itself.

Artificial data. Let us first consider artificial images for which ground truth information is available. For the training data, we generated $N = 1000$ RGB image patches ($F = 3$) of size $\vec{D} = (50,50)$ according to the model (1) to (4). Each patch contained one of five different character types with equal probability ($\pi_c = 0.2$). The chosen colored characters were generated from corresponding mask, mean and variance parameters (see Fig. 1a). The background color was drawn from a Mixture of Gaussians as an example of multi-modal distributions (see Fig. 1a). Fig. 2b shows a random selection of 5 generated data points. The derived EM learning algorithm was applied to the data assuming $C = 5$ classes and $\vec{P} = \vec{D} = (50,50)$. First, the background histogram $\mathcal{H}_B$ was computed from the whole data set, and was observed to model the true generating distributions with high accuracy (the blue regions in Fig. 2a show the learned histograms compared to the true distributions in red). To infer the remaining model parameters they were first initialized: the pattern mean W was independently and uniformly drawn from the RGB-color-cube $[0,1]^3$; the pattern variance Φ was set to the standard deviation of the data set; and the initial mask parameters A were uniformly drawn from the interval $[0,1]$. The learning course of the parameters is illustrated in Fig. 2c with iteration 0 showing the initial values. After iteration 70, parameters had converged sufficiently. To visualize pattern variances in Fig. 2c, they are organized as a matrix for each pattern and each feature dimension, e.g. $\sigma^c_R = (\sigma^c_{(1,1),f=R},\ldots,\sigma^c_{(P_1,P_2),f=R})$. These

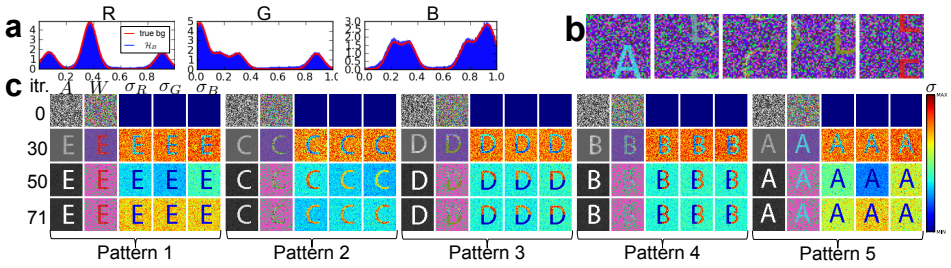

Figure 2. Experiment on artificial data. **a** The background for data generation (red curves) and constructed histogram $\mathcal{H}_B$ (blue regions). **b** 5 of $N = 1000$ image patches. **c** The learning course of the parameters. W is visualized in RGB color space and σ_R, σ_G and σ_B are visualized by heat maps.

variance matrices are visualized by color images which are normalized individually. As can be observed, the algorithm successfully learned the model parameters. For the experiment of Fig. 2 and other similar experiments, the learned parameters diverged from the generating parameters by on average less than 3.0%. Convergence to local optima has only been observed in very few cases (1 of 10 runs).

Scanned text documents. Let us now apply the learning algorithm to data from a single page of a scanned text document. Consider the corrupted document displayed in Fig. 3e which contains 5 character types, "a", "b", "e", "s" and "y". The printed document was manually corrupted with dirt in the form of line-strokes and with grayish spots. The dataset for training was created by a high-resolution scan of the document (3307×4677 pixels) and by automatically cutting the scan into small patches (120×165 pixels) with fixed intervals. Fig. 3a shows five examples of such patches. The patches are used to generate the actual data points $Y^{(n)}$ with vectorial features. Instead of RGB feature vectors as for the introductory example, we used feature vectors generated through Gabor filter responses. Gabor features are robust and widespread in image processing (see, e.g., [5, 13]) with high sensitivity to edge-like structures and textures. Furthermore, they are tolerant w.r.t. small local deformations and brightness changes. For the small patches we computed a Gabor feature with 40 entries at every third pixel, which resulted in 2D arrays of $D_1 \times D_2 = 40 \times 55$ Gabor feature vectors. The learning algorithm was applied to this data set assuming $C = 6$ classes. The pattern mean W was initialized by randomly selecting $C = 6$ patches from the dataset and cutting out a segment of the pattern size at random positions. The remaining parameters were initialized in the same way as for artificial data. To increase computational efficiency we, furthermore, assumed with $\vec{P} = (30, 40)$ a pattern size smaller than the patch size but still larger than the size of any characters. Parameter optimization (44 EM iterations) took about 25 minutes on a cluster with 15 GPUs (GTX 480). Fig. 3b visualizes the inferred parameters after the application of the learning algorithm (see Suppl. for a visualization of the time-course of learning). As can be observed, the algorithm has successfully represented the five character types. They were represented by different classes using parameters for mask, mean features and feature variances. As only five classes are needed to represent all the characters, the algorithm has assigned a pattern averaging other patterns and dirt to one of the classes (class 4). In numerical experiments on this and other documents, classes not representing characters had either much lower values for learned mask parameters (compare Fig. 3b) or much lower values for learned mixing proportions π_c. We exploited this observation to automatically classify character classes (see Suppl. for details). The full learning procedure then consisted of a repetition of the learning algorithm and a selection of one of the results with the highest number of character classes.

Character detection and identification. Based on the learned representation, characters in a given dirty document can now be detected and identified. We screen through the whole document from upper-left to lower-right patch by patch. Our aim is to identify a character within each patch $Y^{(n)}$ and to assign to each match a quality measure, i.e., a measure reporting how well each character matches the learned representation of its class. To identify the position and type of a character in a patch we compute the MAP estimate of the approximate posterior:

$$\begin{aligned}(c^*, \vec{x}^*) &= \operatorname*{argmax}_{c,\vec{x}} \{p(c, \vec{x} \,|\, Y^{(n)}, \Theta)\} \\ &\approx \operatorname*{argmax}_{(c,\vec{x}) \in \mathcal{K}_n} \{q_n(c, \vec{x}\,; \Theta)\},\end{aligned} \tag{15}$$

with $q_n(c, \vec{x}\,; \Theta)$ and $\mathcal{K}_n$ defined as in Sec. 3. In analogy to template matching ([5, 14] and many more) we refer to the result of the MAP estimate (15) as the *match* for the image patch, to $\vec{x}^*$ as the *matched position* and to c^* as the *matched class*. Furthermore, given the patch $Y^{(n)}$ with match $(c^*, \vec{x}^*)$, we define the *quality* of the match as follows,

$$Q(Y^{(n)}, c^*, \vec{x}^*, \Theta) = 1 - \frac{\sum_{\vec{i}=(1,1)}^{(P_1,P_2)} (\alpha_{\vec{i}}^{c^*})^\gamma [\alpha_{\vec{i}}^{c^*} - p(m_{\vec{i}} = 1 \,|\, Y, c^*, \vec{x}^*, \Theta)]^2}{\sum_{\vec{i}'=(1,1)}^{(P_1,P_2)} (\alpha_{\vec{i}'}^{c^*})^\gamma}, \tag{16}$$

where $p(m_{\vec{i}} = 1 \,|\, Y, c^*, \vec{x}^*, \Theta)$ is the posterior distribution of the binary mask (see Eqn. 10). The negative term in (16) is a normalized distance measure between mask parameters and mask posterior probabilities. To provide some intuition, suppose that the mask parameters are binary, i.e., they are either maximally reliable ($\alpha_{\vec{i}}^c = 1$), or maximally unreliable ($\alpha_{\vec{i}}^c = 0$). Then, the quality reveals the percentages of the pattern c^* being matched in the patch. If for instance a patch contains a complete and clean instance of the pattern c^* at position $\vec{x}^*$, $p(m_{\vec{i}} = 1 \,|\, Y, c^*, \vec{x}^*, \Theta)$ is close or equal to one for all reliable features and zero otherwise. This implies that

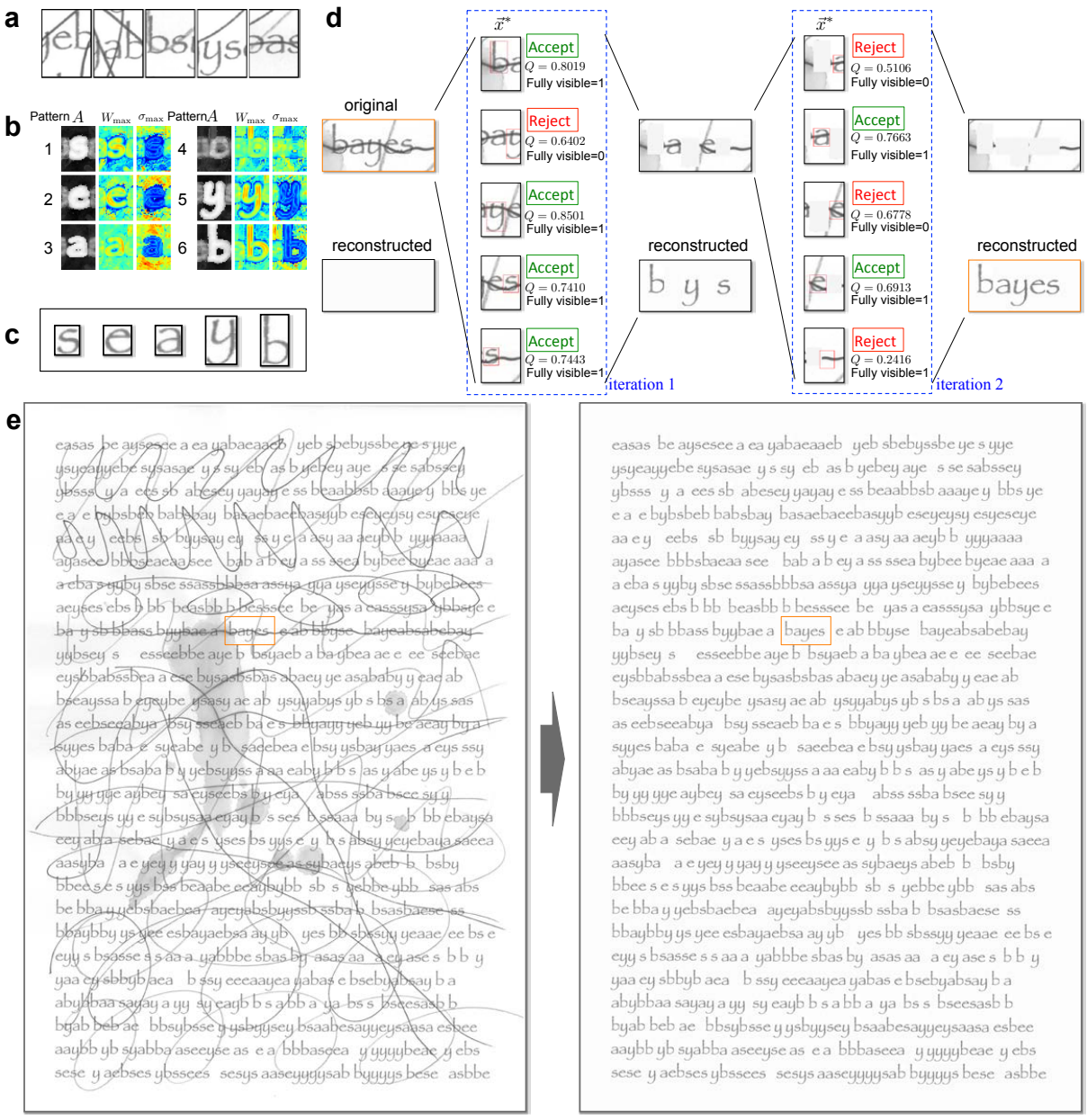

Figure 3. Experiment on text document. **a** 5 of $N = 1379$ image patches. **b** The learned parameters with max representation,e.g., $W^c_{\vec{i},\max} = \max_f(W^c_{\vec{i},f})$ (see Suppl. for the full representation). **c** The clean representation of each character type. **d** An illustration of the cleaning procedure. **e** The cleaning result of our algorithm.

the distance measure is equal to zero and $Q(Y^{(n)}, c^*, \vec{x}^*, \Theta)$ equal to one. For an appropriate scaling of the match quality with the degree of dirt, the unreliable features in (16) have been down-weighted by factors $(\alpha^{c^*}_{\vec{i}})^\gamma$ (we use $\gamma = 10$). Given a patch Y and a match $(c^*, \vec{x}^*)$, the measure (16) thus assigns a quality value $Q(Y^{(n)}, c^*, \vec{x}^*, \Theta) \in [0, 1]$ which well reflects the similarity between an input pattern at $\vec{x}^*$ and its corresponding matched pattern of class c^*. Low values of Q correspond to poor matches and $Q = 1$ corresponds to a perfect match (see Suppl. for details).

5. Corrupted Document Cleaning

By making use of the learned character representation, character matching, and evaluation of match qualities, we can now remove dirt from a given corrupted scanned document. First, we use the match qualities to globally find the best matching input patterns for each class among all extracted patches. For each best input pattern we then com-

pute a bounding box and store the corresponding pixel representation (see Fig. 3c). Then, using the best representations, we can reconstruct the document (see Fig. 3e). In order to do so, we screen through the dirty document patch by patch and for each patch compute the match $(c^*, \vec{x}^*)$ using (15) and the match quality using (16). If the matched position $\vec{x}^*$ corresponds to a pattern fully visible within the patch, and if the match quality is above the threshold $Q_o = 0.5$, we paint the best representation of class c^* at position $\vec{x}^*$ onto an initially blank reconstructed document. Fig. 3d illustrates this procedure for a small area of the example document. As can be observed, not all the matches are accepted for reconstruction, because some matches correspond to patterns not entirely visible (e.g., second patch at iteration 1) or match qualities are too low (e.g., last patch at iteration 2). The quality threshold prevents dirt from being reconstructed as characters. As for each patch just one match is computed, not all characters are reconstructed at first. For a complete reconstruction we therefore replace each successfully reconstructed character in the original document by a blank rectangle (of the same size as the corresponding bounding box) and apply the procedure again. Patterns that previously were not identified because of competition with other patterns can now be found and correctly reconstructed. We terminate the reconstruction once no more matches are accepted. In Fig. 3d two iterations of the procedure are sufficient to successfully reconstruct the word "bayes". The entire document in Fig. 3e is perfectly reconstructed after three iterations. The reconstructions of examples with more character types, non-Latin characters (Klingon) and random placement of characters show similar results (see Supplement). However, the more a document is corrupted by dirt, the less perfect we can expect the reconstruction to be. In examples with dirt fully occluding parts of the document, we do thus obtain many false negative errors (see Supplement). False positive errors are, on the other hand, obtained if, e.g., a random combination of manual line strokes coincides with the feature arrangement of a learned pattern (see Supplement). Although error rates for imperfect reconstructions can be decreased by fine tuning the threshold Q_o, we left the parameter unchanged at $Q_o = 0.5$ for all examples to demonstrate the generality of the approach.

Note that the task of cleaning documents such as those in Fig. 3 or in the Supplement has previously not been addressed. This is because of the difficulty posed by corruptions consisting partly of the same features as the characters (line strokes). Furthermore, extended line strokes severely affect any segmentation-based processing. It is in the nature of a new application domain that no data for comparison is available for our results. To provide, at least, a baseline, we applied a standard OCR approach (FineReader, [15]) to the documents used in our experiments. For the document of Fig. 3, FineReader recognized 56.5% of the characters correctly (essentially those that are segmentable) and corruption by dirt causes 297 false positives. On the same data, our approach detects 100% of the characters correctly with no false positives (FP). More examples can be found in the Supplement. The poorest performance of FineReader in all the examples is observed for documents with non-standard characters or unusual character orientations. For the documents in Figs. 11 and 15 (Suppl. C.2 & C.3), for instance, FineReader results in recognition rates of 0% (231 FP) and 0.8% (86 FP), respectively. For comparison, our approach detects 100% (no FP) in Fig. 11 and 100% (3 FP) in Fig. 15. Performance of the unsupervised learning algorithm is high in these latter two examples because it can learn any character type while the poor performance of FineReader is simply evidence for the data containing characters unknown to the OCR approach. Improvements of OCR would require additional training on labeled data. However, as briefly discussed in the introduction, note that a comparison of OCR to our approach on these data is not fair. OCR is not intended for the task addressed here. Vice versa, our algorithm would not perform well on typical OCR tasks.

6. Discussion

We have studied an unsupervised approach to remove dirt from scanned text documents. Our approach relied on the learning of character representations using a probabilistic generative model with an explicit position variable. Similar to other probabilistic approaches, e.g., image denoising, we followed the general principle of capturing the regularities of the data, and removed unwanted data parts after identifying them as deviations from the learned regularities. However, in contrast to approaches for noise removal, we learned explicit high-level representations of specific image components (characters). Having an explicit notion of feature arrangements per character allows for a discrimination of irregular patterns vs. characters even though these irregular patterns can consist of the same features (line strokes) as the characters themselves. Methods not representing characters explicitly (e.g., [16]) are, therefore, not applicable or would, at the least, require additional mechanisms to identify characters and to discriminate them against irregular patterns.

By applying our approach we have shown in this study that even under difficult conditions a perfect reconstruction of a document is possible with solely the information on a single page. The result of the cleaning procedure depended on the factors like the severity of the corruption, the number of character instances per character type, and on the similarity between character patterns and corrupting patterns. Very simple characters like "I", "V" or "C" are, for instance, easier to confuse with random line strokes than more complex characters. Furthermore, the more character types a docu-

ment contains the more challenging the discrimination between characters becomes, especially for strongly corrupted data. This is true for learning as well as for character identification. Regarding required data, we usually observed good result in our experiments for more than 200 character instances per character type. Performance significantly decreased for less than 100 instances, primarily due to less appropriate learning of the character representations. The example of Fig. 3e contains about 250 instances per character type (1251 characters in total). A page with text consisting of the full alphabet of letters, even if constrained to just lower or upper case, would therefore not provide sufficiently many examples for self-cleaning. A natural extension of the addressed task for more character types would, therefore, require several pages. If we assume that about 200 examples per character type are needed and if a page contains 1000 characters in total, we would require about 6 pages to learn a full Latin alphabet of lower-case letters. For the general type-set of all letters and numbers (excluding special characters), we would require about 13 pages. If we, furthermore, consider that, e.g., just 0.074% of all characters in the English language are of type 'z' [17], then the number of required pages would increase to about 270. To execute the cleaning procedure described in this work, processing of 270 pages amounts to unreasonably long computation times (even using parallel implementations).

On the other hand, the cleaning performance can be further improved by exploiting further regularities of text documents. The regular arrangement of characters along a line (compare [18]) could be used to predict the positions of characters, and linguistic regularities (e.g., probabilistic language models) could be used to predict character types from context. Using probabilistic generative approaches, such prior knowledge can be integrated into the model by constructing more sophisticated prior distribution $p(c, \vec{x} \,|\, \Theta)$. Also on the algorithmic side improvements can certainly be made, e.g., by using a multiple-cause structure (e.g., [19]) to recognize multiple patterns in a patch simultaneously, or by using image features with scale invariance and contrast normalization (e.g., SIFT [20], HOG [21]). Different font sizes of characters can be handled by modeling them as different patterns, adding scaling transformations to the model (dramatically increasing the computational complexity), or estimating font sizes with separate mechanisms.

By applying the probabilistic approach described in this work, we have for the first time shown that it is in principle possible to autonomously clean text documents which are heavily corrupted by irregular patterns. Future developments can further improve the cleaning performance by exploiting regularities of words and sentences, or they can extend the application domain of the approach.

Acknowledgement. This work was funded by the German Research Foundation (DFG) under grant LU 1196/4-1. Early modeling work was funded by the German Federal Ministry of Education and Research (BMBF) under grant 01GQ0840.

References

[1] U. Schmidt, Q. Gao, and S. Roth, "A generative perspective on MRFs in low-level vision," in *CVPR*, 2010. 1

[2] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996. 1, 3

[3] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *NIPS*, pp. 801–808, 2007. 1, 3

[4] B. A. Oishausen, C. H. Anderson, and D. C. V. Essenla, "A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information," *J Neurosci*, vol. 13, pp. 4700–4719, 1993. 1

[5] L. Wiskott, J.-M. Fellous, N. Krger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *PAMI*, vol. 19, pp. 775–779, 1997. 1, 5

[6] B. J. Frey and N. Jojic, "Transformation-invariant clustering using the EM algorithm," *PAMI*, vol. 25, pp. 1–17, 2003. 1, 2

[7] D. B. Grimes and R. P. N. Rao, "Bilinear sparse coding for invariant vision," *Neural Comp*, vol. 17, pp. 47–73, January 2005. 1

[8] C. K. I. Williams and M. K. Titsias, "Greedy learning of multiple objects in images using robust statistics and factorial learning," *Neural Comp*, vol. 16, pp. 1039–1062, 2004. 1, 2, 4

[9] B. J. Frey and N. Jojic, "A comparison of algorithms for inference and learning in probabilistic graphical models.," *PAMI*, vol. 27, pp. 1392–1416, Sept. 2005. 1, 2, 4

[10] J. Winn and A. Blake, "Generative affine localisation and tracking," *NIPS*, 2004. 2

[11] J. Lücke and J. Eggert, "Expectation truncation and the benefits of preselection in training generative models," *JMLR*, vol. 11, pp. 2855 – 2900, 2010. 2, 4

[12] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach Learn*, vol. 37, pp. 183–233, 1999. 4

[13] L. Shen and L. Bai, "A review on Gabor wavelets for face recognition," *PAA*, vol. 9, no. 2-3, pp. 273–292, 2006. 5

[14] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *ISCAS*, pp. 253 – 256, 2010. 5

[15] "Abbyy finereader 11." http://finereader.abbyy.com/. 7

[16] N. Jojic, B. J. Frey, and A. Kannan, "Epitomic analysis of appearance and shape," in *ICCV*, pp. 34–41, 2003. 7

[17] H. Beker and F. Piper, *Cipher Systems: The Protection of Communications*. Wiley-Interscience, 1982. 8

[18] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *PAMI*, vol. 18, pp. 690–706, 1996. 8

[19] P. Dayan and R. S. Zemel, "Competition and multiple cause models," *Neural Comp*, vol. 7, pp. 565 – 579, 1995. 8

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, 2004. 8

[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005. 8